Python IP Class Notebook

Nebhrajani A. V.

Contents

1 Preamble

1.1 Notebook Conventions

All code in this notebook is in Python unless specified otherwise. All code is syntax-highlighted, placed in boxes, and is line numbered. The output of the interpreter on `stdout` is printed directly below it, `verbatim`, thus.

```python
# Print Hello world!
print("Hello world!")
```

```
Hello world!
```

It is recommended that you navigate using the hyperlinked TOC or the Adobe Bookmarks tree.

1.2 Hardware and Software Used

This notebook is written in an `org-mode` file and exported to PDF via LaTeX, Org version 9.3.6 on GNU Emacs 25.2.2 (x86_ 64-pc-linux-gnu, GTK+ Version 3.22.21) of 2017-09-23, modified by Debian, on a Foxconn Core i7 NanoPC running Linux Mint 19.3 XFCE 64-bit. Python 2.7.17 of 2020-04-15 is used throughout unless specified otherwise. For the Org or LaTeX source, contact aditya.v.nebhrajani@gmail.com.

1.3 Acknowledgements

I am grateful to the FSF, the GNU Project, the Linux foundation, the Emacs, StackExchange and FLOSS communities, and my father, who taught me that a world outside commercialized technology does exist and thrive.

2 NumPy

2.1 Worksheet 2020-07-26

1. Create an ndarray with values ranging from 10 to 49 each spaced with a difference of 3.

```
import numpy as np
arr=np.arange(10,50,3,dtype=int)
print(arr)
```

```
[10 13 16 19 22 25 28 31 34 37 40 43 46 49]
```

2. Find the output of the following Python code:

```
x="hello world"
print(x[:2],x[:-2],x[-2:])
```

```
('he', 'hello wor', 'ld')
```

3. Predict the output of the following code fragments:

```
import numpy as np
x=np.array([1,2,3])
y=np.array([3,2,1])
z=np.concatenate([x,y])
print(z)
```

```
[1 2 3 3 2 1]
```

4. Consider following two arrays: Array1= array([0,1,2],[3,4,5],[6,7,8]]) and Array2= array([10,11,12],[13,14,15],[16,17,18]]). Write NumPy command to concatenate Array1 and Array2:

   (a) Row wise

```
import numpy as np
Array1= np.array([[0,1,2],[3,4,5],[6,7,8]])
Array2= np.array([[10,11,12],[13,14,15],[16,17,18]])
rarr=np.concatenate([Array1,Array2],axis=1)
print(rarr)
```

```
[[ 0  1  2 10 11 12]
 [ 3  4  5 13 14 15]
 [ 6  7  8 16 17 18]]
```

(b) Column wise

```
import numpy as np
Array1= np.array([[0,1,2],[3,4,5],[6,7,8]])
Array2= np.array([[10,11,12],[13,14,15],[16,17,18]])
carr=np.concatenate([Array1,Array2],axis=0)
print(carr)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [10 11 12]
 [13 14 15]
 [16 17 18]]
```

5. To create sequences of numbers, NumPy provides a function (a)arange analogous to range that returns arrays instead of lists.

6. Find the output of following program.

```
import numpy as np
a=np.array([30,60,70,30,10,86,45])
print(a[-2:6])
```

```
[86]
```

7. Write a NumPy program to create a 2d array with 1 on the border and 0 inside.

```
import numpy as np
x = np.ones((5,5))
print("Original array:")
print(x)
print("1 on the border and 0 inside in the array")
x[1:-1,1:-1] = 0
print(x)
```

```
Original array:
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
1 on the border and 0 inside in the array
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

8. Given following ndarray A: ([[2, 4, 6], [7, 8, 9], [1, 2, 3]]) Write the python statements to perform the array slices in the way so as to extract first row and second column.

```python
import numpy as np
A = np.array([[2,4,6],[7,8,9],[1,2,3]])
print(A[0,:])
print(A[:,1])
```

```
[2 4 6]
[4 8 2]
```

9. Write python statement to create a two- dimensional array of 4 rows and 3 columns. The array should be filled with ones.

```python
import numpy as np
x = np.ones((4,3))
print(x)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

10. Find the output of following program.

```python
import numpy as np
d = np.array([10,20,30,40,50,60,70])
print(d[-5:])
```

```
[30 40 50 60 70]
```

11. State at least two differences between a NumPy array and a list

NumPy Array	List
By default, numpy arrays are homogeneous Element-wise operations are possible They take up less space	They can have elements of different data types Element-wise operations don't work on lists They take up more space

12. Find the output of following program.

```
import numpy as np
d=np.array([10,20,30,40,50,60,70])
print(d[-1:-4:-1])
```

```
[70 60 50]
```

13. Write the output of the following code.

```
import numpy as np
a=[[1,2,3,4],[5,6,7,8]]
b=[[1,2,3,4],[5,6,7,8]]
n=np.concatenate((a, b), axis=0)
print(n[1])
print(n[1][1])
```

```
[5 6 7 8]
6
```

14. Which of the following is contained in NumPy library?

    (a) **N-Dimensional Array Object**

    (b) Series

    (c) DataFrame

    (d) Plot

15. Point out the correct statement:

    (a) NumPy main object is the homogeneous multidimensional array

    (b) In Numpy, dimensions are called axes

    (c) NumPy array class is called ndarray

    (d) **All of the above**

16. When the fromiter() is preferred over array()? **A:** Fromiter() is preferred over array()for creating non-numeric sequences like strings and dictionaries.

17. What is the purpose of order argument in empty(). What do ‘C’ and ‘F’ stands for? What is the default value of order argument? **A:** The “order” argument arranges the elements of the array row-wise or column-wise. C order arranges elements column wise and means “c”-like, whereas F order arranges elements row wise and means “fortran”-like. Default value of order argument is C.

18. Differentiate split() from hsplit() and vsplit(). **A:** Split() function is a general function which can be used to split an array in numpy both horizontally and vertically by providing an axis. If the axis is 0 it is the same as hsplit() and if the axis is 1 it behaves as vsplit(). The difference between split() and hsplit(),vsplit() is that split() allows you to specify the axis that you wish, and hsplit() and vsplit() are for specific axes.

19. Find the output:

(a)

```
import numpy as np
a = np.linspace(2.5,5,6)
print(a)
```

```
[2.5 3.  3.5 4.  4.5 5. ]
```

(b)

```
import numpy as np
a=np.array([[0,2,4,6],[8,10,12,14],[16,18,20,22],[24,26,28,30]])
print(a)
print(a[:3,3:])
print(a[1::2,:3])
print(a[-3:-1,-4::2])
print(a[::-1,::-1])
```

```
[[ 0  2  4  6]
 [ 8 10 12 14]
 [16 18 20 22]
 [24 26 28 30]]
[[ 6]
 [14]
 [22]]
[[ 8 10 12]
 [24 26 28]]
[[ 8 12]
 [16 20]]
[[30 28 26 24]
 [22 20 18 16]
 [14 12 10  8]
 [ 6  4  2  0]]
```

3 Pandas

3.1 Series

```python
# Import numpy and pandas
import pandas as pd
import numpy as np

# Create an empty series
s = pd.Series()
print(s)

# Series from ndarray
data = np.array(['a', 'b', 'c', 'd'])

## Without index
s = pd.Series(data)
print(s)
## With index
s = pd.Series(data, index = [100, 101, 102, 103])
print(s)

# Scalar series
s = pd.Series(5, index = [0, 1, 2, 3])
print(s)

# Series from dictionary
data = {'a' : 0., 'b' : 1., 'c' : 2.}

## Without index
s = pd.Series(data)
print(s)
## With index
s = pd.Series(data, index = ['b', 'c', 'd', 'a'])
print(s)

# Another dictionary example
f_dict = {'apples': 500, 'kiwi': 20, 'oranges': 100, 'cherries': 6000}
print(f_dict)

arr = pd.Series(f_dict)
print('\nArray Items')
print(arr)
```

```
Series([], dtype: float64)
0    a
1    b
2    c
```

```
3    d
dtype: object
100    a
101    b
102    c
103    d
dtype: object
0    5
1    5
2    5
3    5
dtype: int64
a    0.0
b    1.0
c    2.0
dtype: float64
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
{'kiwi': 20, 'cherries': 6000, 'apples': 500, 'oranges': 100}

Array Items
apples       500
cherries    6000
kiwi          20
oranges      100
dtype: int64
```

```python
# Indexing
import pandas as pd
from pandas import Series
arr = Series([22, 44, 66, 88, 108])
print(arr[[1, 3, 0, 4]])
```

```
1     44
3     88
0     22
4    108
dtype: int64
```

```python
# Series operations
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 9])
print(ds1)
print(ds2)
ds = ds1 + ds2
print("Add two Series:")
print(ds)
print("Subtract two Series:")
ds = ds1 - ds2
print(ds)
print("Multiply two Series:")
ds = ds1 * ds2
print(ds)
print("Divide Series1 by Series2:")
ds = ds1 / ds2
print(ds)
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
0    1
1    3
2    5
3    7
4    9
dtype: int64
Add two Series:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Subtract two Series:
0    1
1    1
2    1
3    1
4    1
dtype: int64
Multiply two Series:
0     2
1    12
2    30
3    56
4    90
dtype: int64
Divide Series1 by Series2:
```

```
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

```python
# Series to array
import pandas as pd
import numpy as np
s1 = pd.Series(['100', '200', '300', 'python'])
print("Original data series")
print(s1)
print("Series to array")
a = np.array(s1.values.tolist())
print(a)
```

```
Original data series
0       100
1       200
2       300
3    python
dtype: object
Series to array
['100' '200' '300' 'python']
```

```python
# Heads and tails
import pandas as pd
import math
s = pd.Series(data = [math.sqrt(x) for x in range(1,10)],
              index = [x for x in range(1,10)])
print(s)
print(s.head(6))
print(s.tail(7))
print(s.head())
print(s.tail())
```

```
1    1.000000
2    1.414214
3    1.732051
4    2.000000
5    2.236068
6    2.449490
7    2.645751
8    2.828427
9    3.000000
```

```
dtype: float64
1    1.000000
2    1.414214
3    1.732051
4    2.000000
5    2.236068
6    2.449490
dtype: float64
3    1.732051
4    2.000000
5    2.236068
6    2.449490
7    2.645751
8    2.828427
9    3.000000
dtype: float64
1    1.000000
2    1.414214
3    1.732051
4    2.000000
5    2.236068
dtype: float64
5    2.236068
6    2.449490
7    2.645751
8    2.828427
9    3.000000
dtype: float64
```

```python
# Sorting pandas series
import pandas as pd
s = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original data series:")
print(s)
asc_s = pd.Series(s).sort_values()
print(asc_s)
dsc_s = pd.Series(s).sort_values(ascending=False)
print(dsc_s)

# Appending
new_s = s.append(pd.Series(['500', 'php']))
print(new_s)
```

```
Original data series:
0       100
1       200
2    python
```

```
3    300.12
4       400
dtype: object
0       100
1       200
3    300.12
4       400
2    python
dtype: object
2    python
4       400
3    300.12
1       200
0       100
dtype: object
0       100
1       200
2    python
3    300.12
4       400
0       500
1       php
dtype: object
```

```python
# Mean and median
import pandas as pd
s = pd.Series(data = [1,2,3,4,5,6,7,8,9,5,3])
print("Original data series:")
print(s)
print("Mean:")
print(s.mean())
print("Standard deviation:")
print(s.std())
```

```
Original data series:
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9     5
10    3
dtype: int64
```

```
Mean:
4.818181818181818
Standard deviation:
2.522624895547565
```

```
# Isin function
import numpy as np
import pandas as pd

s = pd.Series(['dog', 'cow', 'dog', 'cat', 'lion'], name='animal')

r = s.isin(['dog', 'cat'])
print(r)
```

```
0     True
1    False
2     True
3     True
4    False
Name: animal, dtype: bool
```

```
# Appending and concatenation
 import numpy as np
 import pandas as pd

 # Input
 ser1 = pd.Series(range(5))
 ser2 = pd.Series(list('abcde'))

 # Vertical
 ser3 = ser1.append(ser2)
 print(ser3)

 # Or using Pandas concatenate along axis 0
 ser3 = pd.concat([ser1, ser2], axis = 0)
 print(ser3)

 # Horizontal (into a dataframe)
 ser3 = pd.concat([ser1, ser2], axis = 1)
 print(ser3)
```

3.2 Dataframe

```
# Empty dataframe
import pandas as pd

```

```python
data = pd.DataFrame()
print(data)
```

```
Empty DataFrame
Columns: []
Index: []
```

```python
# Dataframe from list
import pandas as pd

table = [1, 2, 3, 4, 5]
data = pd.DataFrame(table)
print(data)
```

```
   0
0  1
1  2
2  3
3  4
4  5
```

```python
# Dataframe from mixed list
import pandas as pd

table = [[1, 'Nebhrajani'], [2, 'Python'], [3, 'Hello']]
data = pd.DataFrame(table)
print(data)
```

```
   0           1
0  1  Nebhrajani
1  2      Python
2  3       Hello
```

```python
# Column labels
import pandas as pd

table = [[1, 'Nebhrajani'], [2, 'Python'], [3, 'Hello']]
data = pd.DataFrame(table, columns = ['S.No', 'Name'])
print(data)
```

```
   S.No        Name
0     1  Nebhrajani
1     2      Python
2     3       Hello
```

```python
# Random numbers dataframe
import numpy as np
import pandas as pd

d_frame = pd.DataFrame(np.random.randn(8, 4))
print(d_frame)
```

```
          0         1         2         3
0 -2.015128  0.313756  0.971252  0.687543
1  0.736305  0.582047  0.171059  0.187363
2  0.983482 -0.118117 -0.548964  0.008497
3  0.739370  0.578854 -0.592963  1.801482
4  0.485355 -0.956433  0.952777  0.297853
5  2.063446 -1.344621 -0.421031  1.592392
6  0.355908  0.432542  1.276645  0.438290
7  1.044685 -0.695084 -0.227355  0.221908
```

```python
# Dataframe from dict
import pandas as pd

table = {'name': ['Aditya', 'Aryan', 'Nebhrajani', 'Sahej'],
        'Salary':[1000000, 1200000, 900000, 1100000]}

data = pd.DataFrame(table)
print(data)
```

```
    Salary        name
0  1000000      Aditya
1  1200000       Aryan
2   900000  Nebhrajani
3  1100000       Sahej
```

```python
# Dataframe from some given dictionary data
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
              'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes',
                    'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

```

```python
df = pd.DataFrame(exam_data , index=labels)
print(df)
```

```
   attempts       name qualify  score
a         1  Anastasia     yes   12.5
b         3       Dima      no    9.0
c         2  Katherine     yes   16.5
d         3      James      no    NaN
e         2      Emily      no    9.0
f         3    Michael     yes   20.0
g         1    Matthew     yes   14.5
h         1      Laura      no    NaN
i         2      Kevin      no    8.0
j         1      Jonas     yes   19.0
```

```python
# Messing with columns
import pandas as pd

table = {'name': ['Aditya', 'Aryan', 'Nebhrajani', 'Sahej'],
         'Age': [25, 32, 30, 26],
         'Profession': ['Developer', 'Analyst', 'Admin', 'HR'],
         'Salary':[1000000, 1200000, 900000, 1100000]
         }

data1 = pd.DataFrame(table)
print(data1)

print('\n After Changing the Column Order')
data2 = pd.DataFrame(table, columns = ['name', 'Profession', 'Salary',
                                       'Age'])
print(data2)
print('\n Using Wrong Column ')
data3 = pd.DataFrame(table, columns = ['name', 'Qualification', 'Salary',
                                       'Age'])
print(data3)
```

```
   Age Profession   Salary        name
0   25  Developer  1000000      Aditya
1   32    Analyst  1200000       Aryan
2   30      Admin   900000  Nebhrajani
3   26         HR  1100000       Sahej

 After Changing the Column Order
         name Profession   Salary  Age
0      Aditya  Developer  1000000   25
1       Aryan    Analyst  1200000   32
```

```
2  Nebhrajani      Admin   900000   30
3       Sahej         HR  1100000   26

 Using Wrong Column
        name Qualification   Salary  Age
0      Aditya           NaN  1000000   25
1       Aryan           NaN  1200000   32
2  Nebhrajani           NaN   900000   30
3       Sahej           NaN  1100000   26
```

```python
# Dataframe indexing
import pandas as pd

table = {'name': ['Aditya', 'Aryan', 'Nebhrajani', 'Sahej'],
         'Age': [25, 32, 30, 26],
         'Profession': ['Developer', 'Analyst', 'Admin', 'HR'],
         'Salary':[1000000, 1200000, 900000, 1100000]
         }
data = pd.DataFrame(table)
print(data)

print('\nSetting name as an index')
new_data = data.set_index('name')
print(new_data)

print('\nReturn Index Aditya Details')
print(new_data.loc['Aditya'])
```

```
   Age Profession   Salary        name
0   25  Developer  1000000      Aditya
1   32    Analyst  1200000       Aryan
2   30      Admin   900000  Nebhrajani
3   26         HR  1100000       Sahej

Setting name as an index
            Age Profession   Salary
name
Aditya       25  Developer  1000000
Aryan        32    Analyst  1200000
Nebhrajani   30      Admin   900000
Sahej        26         HR  1100000

Return Index Aditya Details
Age                  25
Profession    Developer
Salary          1000000
Name: Aditya, dtype: object
```

```python
import pandas as pd

table = {'name': ['Aditya', 'Aryan', 'Nebhrajani', 'Sahej'],
         'Age': [25, 31, 35, 26],
         'Salary':[100000, 120000, 700000, 110000]
            }

data = pd.DataFrame(table)
print(data)
print('\nShape and Size of a DataFrame')
print(data.shape)
data2 = pd.DataFrame(table, columns = ['name', 'Profession', 'Salary',
                                       'Age'])
data3 = pd.DataFrame(table, columns = ['name', 'Qualification', 'Salary',
                                       'Age'])
print('Data2 Values ')
print(data2.values)
print('\nData3 Values ')
print(data3.values)
data1 = pd.DataFrame(table)
table = {'Age': [25, 32, 30, 26],
         'Salary':[1000000, 1200000, 900000, 1100000]
         }
data4 = pd.DataFrame(table)
data1.index.name = 'Emp No'
print(data1)
print()
data4.index.name = 'Cust No'
print(data4)
data1.columns.name = 'Employee Details'
print(data1)
data4.columns.name = 'Customers Information'
print(data4)
data1 = pd.DataFrame(table)
print(data1)
print('\nDescribe function result')
print(data1.describe())
```

```
   Age  Salary        name
0   25  100000     Aditya
1   31  120000      Aryan
2   35  700000  Nebhrajani
3   26  110000      Sahej

Shape and Size of a DataFrame
(4, 3)
```

```
Data2 Values
[['Aditya' nan 100000 25]
 ['Aryan' nan 120000 31]
 ['Nebhrajani' nan 700000 35]
 ['Sahej' nan 110000 26]]

Data3 Values
[['Aditya' nan 100000 25]
 ['Aryan' nan 120000 31]
 ['Nebhrajani' nan 700000 35]
 ['Sahej' nan 110000 26]]
        Age  Salary        name
Emp No
0        25  100000      Aditya
1        31  120000       Aryan
2        35  700000  Nebhrajani
3        26  110000       Sahej
()
         Age   Salary
Cust No
0         25  1000000
1         32  1200000
2         30   900000
3         26  1100000
Employee Details  Age  Salary        name
Emp No
0                  25  100000      Aditya
1                  31  120000       Aryan
2                  35  700000  Nebhrajani
3                  26  110000       Sahej
Customers Information  Age   Salary
Cust No
0                       25  1000000
1                       32  1200000
2                       30   900000
3                       26  1100000
   Age   Salary
0   25  1000000
1   32  1200000
2   30   900000
3   26  1100000

Describe function result
             Age        Salary
count   4.000000  4.000000e+00
mean   28.250000  1.050000e+06
std     3.304038  1.290994e+05
min    25.000000  9.000000e+05
```

```
25%    25.750000  9.750000e+05
50%    28.000000  1.050000e+06
75%    30.500000  1.125000e+06
max    32.000000  1.200000e+06
```